



6^{ex}
1-24-01

1807.1033

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:)
SYLVAIN JOYEAU ET AL.)
Application No.: 09/882,301)
Filed: June 18, 2001)
For: DEVICE AND METHOD FOR)
CONTROLLING ACCESS TO)
COMPUTER PERIPHERALS)
Examiner: N.Y.A.
Group Art Unit: 2185
October 10, 2001

RECEIVED
OCT 17 2001
Technology Center 2100

Commissioner for Patents
Washington, D.C. 20231

CLAIM TO PRIORITY

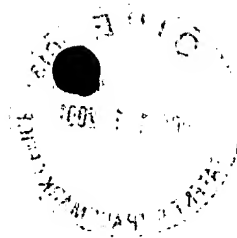
Sir:

Applicants hereby claim priority under the International Convention and all
rights to which they are entitled under 35 U.S.C. § 119 based upon the following French

Priority Application:

0007751, filed June 16, 2000.

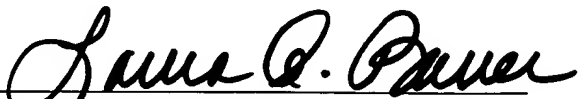
A certified copy of the priority document is enclosed.



THIS PAGE BLANK (USPTO)

Applicants' undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,


Attorney for Applicants

Registration No. 29,767

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

NY_MAIN197620v1

THIS PAGE BLANK (USPTO)



RECEIVED
OCT 17 2001
Technology Center 2100

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le **02 MAI 2001**

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersburg
75800 PARIS cedex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30
<http://www.inpi.fr>

THIS PAGE BLANK (USPTO)

REQUÊTE EN DÉLIVRANCE 1/2

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 W / 260899

| | | | |
|---|----------------------|---|--|
| <p>REMISE DES PIÈCES</p> <p>DATE 16 JUIN 2000</p> <p>LIEU 75 INPI PARIS</p> <p>N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0007751</p> <p>DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 16 JUIN 2000</p> | | <p>1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE</p> <p>• RINUY, SANTARELLI 14, avenue de la Grande Armée 75017 PARIS</p> | |
| <p>Vos références pour ce dossier (facultatif) BIF022115/FR</p> | | | |
| <p>Confirmation d'un dépôt par télécopie</p> | | <p><input type="checkbox"/> N° attribué par l'INPI à la télécopie</p> | |
| <p>2 NATURE DE LA DEMANDE</p> | | <p>Cochez l'une des 4 cases suivantes</p> | |
| Demande de brevet | | <input checked="" type="checkbox"/> | |
| Demande de certificat d'utilité | | <input type="checkbox"/> | |
| Demande divisionnaire | | <input type="checkbox"/> | |
| <p><i>Demande de brevet initiale</i></p> <p><i>ou demande de certificat d'utilité initiale</i></p> | | <p>N° _____ Date ____/____/____</p> <p>N° _____ Date ____/____/____</p> | |
| Transformation d'une demande de brevet européen <i>Demande de brevet initiale</i> | | <p><input type="checkbox"/></p> <p>N° _____ Date ____/____/____</p> | |
| <p>3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)</p> <p>Dispositif et procédé de contrôle d'accès de périphérique informatique.</p> | | | |
| <p>4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE</p> | | <p>Pays ou organisation _____ N° _____</p> <p>Date ____/____/____</p> <p>Pays ou organisation _____ N° _____</p> <p>Date ____/____/____</p> <p>Pays ou organisation _____ N° _____</p> <p>Date ____/____/____</p> <p><input type="checkbox"/> S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»</p> | |
| <p>5 DEMANDEUR</p> | | <p><input type="checkbox"/> S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»</p> | |
| Nom ou dénomination sociale | | CANON KABUSHIKI KAISHA | |
| Prénoms | | | |
| Forme juridique | | Société de droit Japonais | |
| N° SIREN | | | |
| Code APE-NAF | | | |
| Adresse | Rue | 30-2, Shimomaruko 3-chome, Ohta-ku, | |
| | Code postal et ville | Tokyo | |
| Pays | | JAPON | |
| Nationalité | | JAPONAISE | |
| N° de téléphone (facultatif) | | | |
| N° de télécopie (facultatif) | | | |
| Adresse électronique (facultatif) | | | |

| | | | | |
|--|----------------------|--|--------------|-------------------|
| REMISE DES PIÈCES DATE 16 JUIN 2000 LIEU 75 INPI PARIS N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0007751 | | Réservé à l'INPI | | DB 540 W / 260899 |
| Vos références pour ce dossier : <i>(facultatif)</i> | | BIF022115/FR | | |
| 6 MANDATAIRE | | | | |
| Nom | | | | |
| Prénom | | | | |
| Cabinet ou Société | | RINUY, SANTARELLI | | |
| N° de pouvoir permanent et/ou de lien contractuel | | | | |
| Adresse | Rue | 14 AVENUE DE LA GRANDE ARMÉE | | |
| | Code postal et ville | 750017 | PARIS | |
| N° de téléphone <i>(facultatif)</i> | | 01 40 55 43 43 | | |
| N° de télécopie <i>(facultatif)</i> | | | | |
| Adresse électronique <i>(facultatif)</i> | | | | |
| 7 INVENTEUR (S) | | | | |
| Les inventeurs sont les demandeurs | | <input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non Dans ce cas fournir une désignation d'inventeur(s) séparée | | |
| 8 RAPPORT DE RECHERCHE | | Uniquement pour une demande de brevet (y compris division et transformation) | | |
| Établissement immédiat ou établissement différé | | <input checked="" type="checkbox"/> <input type="checkbox"/> | | |
| Paiement échelonné de la redevance | | Paiement en deux versements, uniquement pour les personnes physiques <input type="checkbox"/> Oui <input type="checkbox"/> Non | | |
| 9 RÉDUCTION DU TAUX DES REDEVANCES | | Uniquement pour les personnes physiques <input type="checkbox"/> Requête pour la première fois pour cette invention <i>(joindre un avis de non-imposition)</i> <input type="checkbox"/> Requête antérieurement à ce dépôt <i>(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence)</i> : | | |
| Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes | | | | |
| 10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) | | VISA DE LA PRÉFECTURE OU DE L'INPI | | |
| Bruno QUANTIN N°92.1206 RINUY, SANTARELLI | |  | | |

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08

Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

DÉSIGNATION D'INVENTEUR(S) Page N° 1. / 1.

(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 W / 260899

| | | | |
|--|-----------------------------|---|------------------------------|
| Vos références pour ce dossier (facultatif) | | BIF022115/FR | |
| N° D'ENREGISTREMENT NATIONAL | | 0007751 | |
| TITRE DE L'INVENTION (200 caractères ou espaces maximum) | | | |
| Dispositif et procédé de contrôle d'accès de périphérique informatique. | | | |
| LE(S) DEMANDEUR(S) : CANON KABUSHIKI KAISHA | | | |
| DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inventeurs, utilisez un formulaire identique et numérotez chaque page en indiquant le nombre total de pages). | | | |
| Nom | | JOYEAU | |
| Prénoms | | Sylvain | |
| Adresse | Rue | 77 Avenue Aristide Briand | |
| | Code postal et ville | 35000 | RENNES, France. |
| Société d'appartenance (facultatif) | | | |
| Nom | | ABIVEN | |
| Prénoms | | Anne | |
| Adresse | Rue | 14, rue de la Grange | |
| | Code postal et ville | 35510 | CESSON SEVIGNE CEDEX, France |
| Société d'appartenance (facultatif) | | | |
| Nom | | SANCHEZ-LEIGHTON | |
| Prénoms | | Vicente | |
| Adresse | Rue | 19 rue de la Fonderie | |
| | Code postal et ville | 35000 | RENNES, France. |
| Société d'appartenance (facultatif) | | | |
| DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire) | | Le 16 juin 2000 Bruno QUANTIN N°92.1206 RINUY, SANTARELLI | |

THIS PAGE BLANK (USPTO)

5

10 L'invention est du domaine général des dispositifs informatiques.
Elle concerne en particulier un dispositif de contrôle de périphérique par des applications exécutées sur un système informatique multi-applicatif.

Dans les systèmes multi-applicatifs actuels, les applications ont accès à un périphérique à travers une couche logicielle appelée pilote de
15 périphérique. Ce pilote permet d'exporter une vue abstraite et générique (pour un système d'exploitation donné) d'un périphérique donné.

D'un coté, cette abstraction permet de simplifier la conception, le développement et le portage des applications. D'un autre coté, les communications entre plusieurs applications et le périphérique sont centralisées
20 et coordonnées par le pilote du périphérique. Ceci permet de partager un périphérique entre plusieurs applications, tout en gardant l'intégrité du système.

En contre partie, cette vue abstraite du périphérique induit un handicap majeur dans un système où un niveau de performance élevé est nécessaire. De plus, compte tenu de l'abstraction du périphérique par le pilote,
25 les applications ne peuvent pas exploiter entièrement les capacités particulières des périphériques, puisque les applications doivent se plier au protocole de communication avec le pilote. Les applications ne peuvent pas communiquer directement avec l'interface du périphérique.

Pour tenter de résoudre ces problèmes plusieurs solutions
30 ponctuelles ont été proposées.

On connaît tout d'abord le mécanisme d'accès direct en mémoire (Direct Memory Access, ou DMA) pour le transfert de données entre la mémoire

centrale et un périphérique donné sans intervention de l'unité centrale (CPU).

Ce mécanisme permet à une application de spécifier l'adresse d'un bloc de mémoire tampon (buffer) ainsi que la taille impliquée dans la prochaine commande.

5 Or, d'une part, assez généralement, les périphériques courants ne peuvent utiliser une adresse que sous forme physique (et non virtuelle), et, d'autre part, dans un système multi-applicatif, le système d'exploitation met en place un gestionnaire de mémoire virtuelle (MMU, pour "Memory Management Unit" en terminologie anglosaxonne) afin de commuter plus facilement entre les
10 espaces mémoires des applications.

De ce fait, dans un système multi-applicatif standard, chaque requête impliquant un mécanisme d'accès en mémoire direct (DMA) passe donc par le système d'exploitation (OS) pour convertir les adresses virtuelles fournies par l'application en adresses physiques compréhensibles par le
15 périphérique impliqué. Le système d'exploitation est à nouveau utilisé.

Un procédé, décrit dans le brevet US 5 659 798 ("Method and system for initiating and loading DMA controller registers by using user-level programs", Blumrich) ajoute quelques fonctionnalités au système d'exploitation et un mécanisme matériel associé pour éviter le système d'exploitation dans la
20 programmation par accès direct en mémoire (DMA). Il utilise particulièrement un module de décodage d'adresses inséré sur le bus système, et une initialisation particulière de l'espace virtuel de chaque application. L'application peut alors fournir directement les adresses physiques et les tailles des blocs de mémoire tampon (buffers), tout en garantissant l'intégrité du système.

25 Ce procédé pour éviter l'accès au système d'exploitation est cependant limité à la programmation du mécanisme d'accès direct à la mémoire, et ne permet nullement d'accéder à la programmation de l'ensemble de l'interface d'un périphérique.

On peut également citer le protocole I2O, utilisant un processeur
30 spécifique d'entrées-sorties (Input Output Processor, IOP), placé entre la mémoire centrale et les périphériques, et destiné à décharger l'unité centrale (CPU) du traitement des interruptions lors des traitements haut débit.

Dans ce protocole, on développe un pilote logiciel commun à toute une classe de périphériques (par exemple toutes les interfaces réseau), spécifique à un système d'exploitation. Chaque constructeur de périphérique de cette classe développe alors un pilote logiciel spécifique qui s'exécutera sur le processeur IOP.

On constate bien alors une réduction du traitement des tâches spécifiques à une classe particulière de périphériques par le système d'exploitation. Toutefois, cela est réalisé au prix d'une augmentation du niveau d'abstraction de l'accès aux périphériques par les applications (avec des commandes de contrôle d'un seul pilote commun à toute une classe de périphériques).

Le problème formulé ici est alors de fournir un accès direct sans abstraction à différentes applications exécutées sur un système multi-applicatif, tout en garantissant l'intégrité du système.

L'invention vise en premier lieu un dispositif de partage et de contrôle d'accès de périphérique pour un système informatique comprenant un processeur central (CPU) et au moins un périphérique d'entrée-sortie comportant une interface physique de contrôle accessible au processeur central,

caractérisé en ce que ledit dispositif comporte :

des moyens de reproduction fidèle sous forme d'interface virtuelle de l'interface physique d'au moins un périphérique,

des moyens d'interception par ladite interface virtuelle de toutes les requêtes et données échangées entre le processeur central et le périphérique, commandées par une application prédéterminée exécutée dans le système,

des moyens de modification éventuelle desdites requêtes et données interceptées selon au moins un critère prédéterminé.

On comprend que par cette disposition, les applications exécutées sur le système peuvent accéder aux périphériques de façon directe, sans passer par l'intermédiaire de l'unité de pilotage, et en choisissant donc un niveau d'abstraction des commandes adapté à leurs besoins. Il est alors

possible pour chaque application de spécialiser la programmation du périphérique pour utiliser au mieux ses performances en fonction du résultat souhaité, spécifique à la dite application.

La création d'une interface virtuelle reproduisant presque à l'identique l'interface physique du périphérique permet de réaliser des fonctions de filtre d'accès en lecture ou en écriture du périphérique, et donc de conserver une vérification d'intégrité du système (c'est-à-dire isoler une erreur survenue dans une application particulière sans affecter les autres applications). L'interface virtuelle reproduit un sous-ensemble maximal de l'interface physique. Elle permet ainsi d'exploiter de la même manière les fonctions principales proposées par le périphérique.

Pour simplifier, ce mécanisme peut être vu comme une transposition du mécanisme de mémoire virtuelle vers les accès aux périphériques, avec :

- une fonctionnalité de filtrage en plus,
- une granularité d'accès beaucoup plus fine.

Rappelons que dans le cas du mécanisme de mémoire virtuelle, la granularité de l'espace protégé est appelé la page (terme connu de l'homme du métier), une page étant une zone de mémoire de 4,8,16, 64Ko ou bien plus. On parle d'une page virtuelle lorsqu'elle est vue par l'application. Une page physique réside dans la mémoire physique du système (RAM) ou bien sur le disque. Le mécanisme de mémoire virtuelle associe les pages virtuelles des applications à des pages physiques, chaque page virtuelle étant affectée d'un jeu d'attributs permettant de qualifier les droits d'accès à la page virtuelle par l'application. Dans le cas du Pentium, par exemple, une page peut être soit en lecture seulement, soit en lecture et exécution, soit en lecture et en écriture ou inaccessible.

Dans le mécanisme présenté, la granularité de l'entité protégée est le registre de l'interface physique, soit 8, 16, 32, 64 bits ou plus. Nous appellerons cette entité élémentaire une io-page : vue de l'application, les io-pages sont virtuelles, vue du reste du système, les io-pages sont physiques. Par exemple, le registre de programmation de

DMA noté 135 sur la figure 7 qui sera décrite ultérieurement est une io-page virtuelle, alors que le registre implémenté dans l'interface physique de l'interface notée 9A est une io-page physique notée 136. Comme dans le mécanisme de mémoire virtuelle, l'application peut se voir attribuer, dans son espace d'adressage, un certain nombre d'io-pages virtuelles, dont les accès sont fonctionnellement protégés grâce au mécanisme présenté. On comprend que la virtualisation d'une interface physique, composée d'un jeu propre au périphérique de registres physiques, ou io-pages physiques, passe par l'association d'io-pages physiques à des io-pages virtuelles dans l'espace mémoire adressable de l'application. Comme dans le cas de la mémoire virtuelle, l'association de chaque io-page virtuelle avec son io-page physique est affectée d'attributs permettant le contrôle de l'intégrité du système lors d'accès à l'io-page physique par plusieurs applications. En effet, qualifiant chaque io-page virtuelle, ces attributs permettent de modifier la donnée lue ou écrite par l'application à travers son io-page virtuelle. En plus des droits d'autorisation d'accès en lecture et/ou en écriture, ces attributs contiennent un masque de bits. Chaque donnée écrite par l'application dans une io-page virtuelle, autorisée par ses attributs à être modifiée, est transmise à l'io-page physique en passant par le masque. Les données parcourant le chemin inverse, depuis l'io-page physique vers l'io-page virtuelle d'une application, passent aussi par le masque.

Selon une disposition préférée, les moyens de reproduction sous forme d'io-pages virtuelles d'une interface physique de périphérique comprennent :

- un espace de mémoire virtuelle réservé à l'image de l'interface physique, propre à chaque application exécutée par le système informatique, contenant les io-pages virtuelles de l'application,
- un moyen permettant de lier les adresses de ces espaces de mémoire virtuelle à l'adresse de l'interface physique, contenant les io-pages physiques.

Selon une mise en œuvre particulière, les moyens d'interception

comprennent :

- d'une part une interface avec le bus relié à l'unité centrale de traitement, et une interface avec le bus relié aux périphériques,
- et, d'autre part, un moyen de décodage d'adresses.

5 Le moyen de décodage d'adresses permet de déterminer quels accès doivent être modifiés pour tenir compte d'un critère d'intégrité du système.

 Selon une mise en œuvre plus particulière, les moyens de modification comprennent un moyen de filtrage des requêtes interceptées par
10 les moyens d'interception, selon au moins un critère mémorisé dans un moyen de mémoire.

 Selon une mise en œuvre encore plus particulière, le moyen de filtrage est incorporé à un dispositif mémoire modifiable.

 Selon un premier mode de réalisation, principalement matériel, le
15 dispositif se compose, d'une part, d'un module inséré entre l'unité centrale de traitement et le bus des périphériques et, d'autre part, d'un élément logiciel stocké préalablement dans un dispositif de mémoire de l'unité centrale de traitement, et exécuté par le système d'exploitation à l'initialisation du système.

 L'invention vise plus généralement un téléphone, un appareil
20 photographique, une imprimante, un scanner, une caméra, un ordinateur, un télécopieur, un téléviseur, un lecteur audio/vidéo, caractérisés en ce que ces appareils de traitement de données comportent un dispositif tel qu'exposé brièvement ci-dessus.

 L'invention vise également un moyen de stockage d'informations
25 et un moyen de stockage d'informations amovible, partiellement ou totalement, lisibles par un ordinateur ou un microprocesseur conservant des portions de code d'un programme d'ordinateur, permettant la mise en œuvre du procédé exposé succinctement ci-dessus.

 L'invention vise en outre un produit programme d'ordinateur
30 chargeable dans un appareil programmable, comportant des portions de code logiciel permettant de mettre en œuvre les étapes du procédé tel que brièvement exposé ci-dessus, lorsque le programme est exécuté sur un

appareil programmable.

La description et les dessins qui suivent permettront de mieux comprendre les buts et avantages de l'invention. Il est clair que cette description est donnée à titre d'exemple, et n'a pas de caractère limitatif. Dans

5 les dessins :

- la figure 1 représente sous forme de schéma synoptique un système informatique de type classique ;

10 - le figure 2 représente de façon schématique l'architecture logicielle exécutée sur un système informatique conforme à celui représenté sur la figure 1 ;

- le figure 3 représente fonctionnellement la place du dispositif selon l'invention dans le système informatique ;

- le figure 4 représente sous forme de schéma fonctionnel, une réalisation matérielle du dispositif et du procédé selon l'invention ;

15 - la figure 5 représente un exemple d'espace d'adressage dans un système informatique où résident deux applications partageant un même périphérique ;

- la figure 6 est un organigramme d'un diagramme d'exécution du procédé selon l'invention ;

20 - la figure 7 donne un exemple de mécanisme utilisé pour la mise en œuvre directe d'un moteur de DMA par l'application, sans intervention du système d'exploitation ;

25 - la figure 8 illustre sous forme de schéma fonctionnel, une variante de réalisation de type hybride matérielle et logicielle du dispositif et du procédé selon l'invention ;

De façon générale, l'invention trouve une application dans un système informatique de type classique, tel que par exemple celui illustré sur la figure 1, sous forme de schéma synoptique.

30 Un tel système informatique comporte sur une carte de traitement 1, reliés entre eux par un bus d'adresses et de données 2 :

- 1 - une unité centrale de traitement 3 par exemple de type Pentium (marque déposée de la société Intel);

2 - une mémoire vive RAM 4 ;

3 - une mémoire morte ROM 5 ;

4 - un certain nombre de périphériques P reliés au bus d'adresses et de données 2 par des interfaces P_A , comportant une mémoire interne P_B , ces
5 périphériques comprenant notamment :

- un écran 6 avec une interface d'écran 6A et une mémoire interne non représentée ;

- un clavier 7 avec une interface de clavier 7A et une mémoire interne non représentée;

10 - un lecteur de CD-Roms 8 avec une interface de lecteur de CD-Roms 8A et une mémoire interne non représentée;

- une unité de disque dur 9 avec une interface de disque dur 9A et une mémoire interne non représentée;

15 - un réseau 10 avec une interface réseau 10A et une mémoire interne non représentée;

- un lecteur de disquettes 11 avec une interface de lecteur de disquettes 11A et une mémoire interne non représentée.

20 Chacun des éléments illustrés en figure 1 est bien connu de l'homme du métier des systèmes de traitement de l'information. Ces éléments connus en soi ne sont donc pas décrits ici.

La mémoire vive 4 conserve des données, des variables et des résultats intermédiaires de traitement, dans des zones de mémoire portant, dans la description, les mêmes noms que les données dont ils conservent les valeurs.

25 La mémoire morte 5 est adaptée à conserver par exemple, dans des zones qui, par commodité, possèdent les mêmes noms que les données qu'ils conservent :

- le programme de fonctionnement de l'unité centrale de traitement 3, dans une zone "*program*".

30 L'unité centrale de traitement 3 est adaptée à mettre en œuvre le procédé de contrôle d'accès de périphérique informatique, qui va être exposé ci-dessous.

L'invention s'applique notamment au niveau de la gestion des accès aux périphériques 6 à 11 par des applications logicielles, c'est à dire pratiquement au niveau du bus de communication 2 (par exemple de type PCI dans la suite de la description) entre le processeur de l'unité centrale de traitement 3 sur lequel sont exécutés un système d'exploitation et des applications, et les périphériques 6, 7, 8, 9, 10, 11 (on utilisera également dans la suite de la description le terme de ressource conforme à l'usage courant, pour désigner des périphériques).

Ainsi qu'illustré par la figure 2, une interface de périphérique comporte à la fois une partie matérielle et une partie logicielle.

La partie matérielle est constituée d'une carte "fille" enfichée sur la carte principale ou bien intégrée dans un des circuits de silicium solidaires de la carte principale, par exemple l'interface écran 6A, l'interface réseau 10A, l'interface disque 9A.

La partie logicielle associée est un programme dit pilote de périphérique, spécifique à chaque périphérique, par exemple ici un pilote d'écran 6', un pilote de carte réseau 10', un pilote d'unité de disque dur 9'. Ces pilotes font classiquement partie du système d'exploitation 12, exécuté par l'unité centrale de traitement 3.

La partie logicielle 15 du système informatique comporte enfin des applications, telles que par exemple un logiciel de navigation internet 13, un logiciel d'animation vidéo 14. Ces applications 13, 14 sont également exécutées par l'unité centrale de traitement 12 et adressent de multiples requêtes successives en lecture ou en écriture aux différents périphériques dont elles ont besoin à un instant donné : écran 6, réseau 10, unité de disque dur 9.

Ces applications sont exécutées en parallèle dans un système multi-applicatif.

Avant de décrire plus avant le dispositif et le procédé selon l'invention, on rappelle des informations relatives aux registres de périphériques de systèmes informatiques.

D'une manière générale, l'ensemble, tel que celui noté 9A sur la figure 1, des registres de programmation d'un périphérique, tel que celui noté 9

sur la figure 1, peuvent se ranger en trois groupes :

1/ les registres d'état RE: ils permettent de connaître l'état du périphérique. Par exemple, ils comportent une information signalant que le périphérique a terminé la dernière requête qu'il a reçue, ou qu'il est prêt à en
5 traiter une autre.

Ce type de registre d'état RE est destiné à être uniquement lu par une application, sans qu'il n'y ait jamais d'opération d'écriture par l'application dans ce registre d'état RE.

2/ les registres de paramètres RP : ces registres RP sont
10 accessibles à une application 13, 14 tant en lecture qu'en écriture.

Ils permettent de faire connaître au périphérique 9, 10 les différents paramètres d'une requête qui va suivre. Par exemple, ces paramètres incluent le numéro de cylindre, de secteur et de tête dans le cas d'un contrôleur de disque 9A, ou l'adresse d'écriture des prochains paquets lors de la réception
15 d'une carte réseau 10A.

La lecture comme l'écriture dans ces registres RP ne déclenche aucun processus particulier au niveau du périphérique lui-même, excepté la mémorisation de paramètres.

3/ les registres de contrôle RC : ces registres RC ne sont
20 généralement utiles qu'en écriture.

En effet, c'est en écrivant dans un registre de contrôle RC que l'ordre de déclenchement d'une requête est transmis au périphérique.

La valeur transmise au périphérique par l'écriture dans ce registre de contrôle peut contenir un dernier paramètre décisif, comme le type de requête (demande de transmission, demande de réception etc.), mais la valeur
25 transmise peut aussi être arbitraire, et, dans ce cas, c'est le simple fait d'écrire dans un tel registre RC qui déclenche la commande du périphérique.

La définition de ces trois types de registres de périphérique conduit logiquement à la définition de modèles de protection que doit garantir le
30 système d'exploitation vis-à-vis de la manipulation de ces registres RE, RP, RC par des applications 13, 14. On peut alors distinguer quatre types de protection principaux :

(A) Garantie de l'intégrité des données ainsi que du code des autres applications en cours d'exécution.

Typiquement, le système d'exploitation 12 doit garantir, dans le cas d'accès à un périphérique 9, qu'une application 13 ne puisse pas écraser
 5 une zone mémoire (dans la mémoire vive 4 du système) d'une autre application 14 lors de la réception de données.

Cette contrainte peut être assurée en filtrant les adresses mémoires, ou plus généralement les paramètres, fournis par l'application 13 au
 périphérique 9.

10 **(B) Garantie que la manipulation du périphérique par une application donnée ne perturbe pas l'interaction du périphérique avec les autres applications.**

On comprend que, par exemple, une application 13 ne doit pas pouvoir émettre un ordre de remise à zéro général vers le périphérique, alors
 15 que d'autres applications 14 ne sont pas préparées à une telle remise à zéro.

Cette contrainte peut être assurée en filtrant les ordres de contrôle envoyés par chaque application 13, 14.

(C) Garantie que le résultat de la lecture d'un registre d'état n'induit pas en erreur une application.

20 Il est par exemple clair que l'information signalant que la dernière requête a été exécutée (un bit dans un registre d'état RE) ne doit pouvoir être lue que par l'application 13 responsable de cette requête particulière. Cependant, cette information ne doit pas être lue par une autre application 14 qui vient d'envoyer une nouvelle requête n'ayant pas encore été interprétée par
 25 le périphérique 9.

Par contre, une information indiquant que le périphérique 9 est disponible et prêt à traiter une nouvelle commande doit être accessible à toutes les applications 13, 14.

30 La lecture des registres d'état RE ne doit donc être que partiellement filtrée.

(D) Garantie de distribution correcte des données destinées aux différentes applications.

Si l'on prend ici l'exemple de la gestion d'un périphérique de type contrôleur de disque 9A, on constate que, lors de la lecture de données sur un disque 9 par une application 13, les données d'un secteur sont récupérées par l'application 13 en les lisant les unes après les autres dans un port d'entrée-sortie spécifique de l'interface 9A.

Après chaque lecture, la donnée suivante est disponible sur le port. Si une seconde application 14 est autorisée à lire ce port, la première application 13 génératrice de l'ordre de lecture va subir la perte d'une partie au moins des données qu'elle doit lire, ces données étant lues par la seconde application 14.

Pour un périphérique du type contrôleur de disque 9A, le système d'exploitation 12 doit donc filtrer les accès en lecture à ce type de registre de données.

Le dispositif selon l'invention, dont une configuration générale sous forme de réalisation matérielle est illustrée sur la figure 3, se compose d'un module 16 inséré logiquement entre le couple formé par l'unité centrale de traitement 3 et la mémoire vive 4, d'une part, et le bus 2 des périphériques 6, 9, 10, d'autre part (il s'agit donc en quelque sorte d'un composant servant d'interface entre le processeur et le bus PCI).

Pour faciliter la compréhension de la suite de la description, on a également représenté ici la mémoire cache 17 de l'unité centrale de traitement 3, reliée à cette dernière par un bus système 18. On a aussi représenté un pont 19 permettant d'arbitrer des accès concurrents à la mémoire vive 4 et qui est relié à ladite mémoire par un bus mémoire 20. Ces éléments sont connus en soi de l'homme du métier.

Fonctionnellement, le module 16 décrit ici à titre d'exemple non limitatif comporte alors successivement :

- une interface de bus entrée-sortie 21 reliée par le bus d'entrée/sortie 2' au couple formé de l'unité centrale de traitement 3 et de la mémoire 4 par l'intermédiaire du pont 19,
- une unité de logique programmable 22,
- une interface de bus entrée-sortie 23 reliée au bus d'adresses et

de données 2.

L'unité de logique programmable 22 (par exemple réalisée sous forme d'un ASIC) intercepte tous les accès en lecture et en écriture sur certains périphériques partagés, et peut modifier les données transférées lors de ces
 5 accès, suivant un schéma préprogrammé, défini lors de l'initialisation du système informatique (à la mise en route du système informatique), par le système d'exploitation 12.

L'unité de logique programmable 22 du module 16 comporte notamment :

- 10 - un champ de vecteurs accessibles dans l'espace physique du processeur central 3 dans une zone 160 (figure 5) et permettant de lier les adresses des io-pages virtuelles aux io-pages physiques,
- une mémoire locale 25 du module 16 comportant notamment des champs de bits 132, 142 (figure 5) spécifiant les motifs
 15 de filtrage pour les applications installées 13, 14.

On comprend qu'ainsi le système d'exploitation 12 peut mettre à la disposition d'applications 13, 14, spécifiquement modifiées pour tirer parti de l'invention, un accès direct aux périphériques reliés au module 16, tout en garantissant l'intégrité du système d'exploitation 12 (c'est à dire en particulier en
 20 vérifiant les conditions (A) à (D) définies plus haut). Ceci s'explique par le fait que toutes les données transitant entre les applications 13, 14 et les périphériques passent par l'unité de logique programmable 22.

Plus précisément, l'unité de logique programmable 22 se compose (voir figure 4) d'un décodeur d'adresses 24, d'une mémoire locale 25 et d'un
 25 filtre programmable 26, disposés par exemple sur une carte électronique 27, aux côtés des interfaces de bus entrée-sortie 21, 23.

Le module 16 est inséré en série sur le bus PCI (dans le cas d'un bus de type PCI) connecté à la mémoire vive 4.

L'interface de bus entrée-sortie 21 reliée à l'unité centrale de
 30 traitement 3 envoie au décodeur d'adresses 24 les adresses comprises dans les requêtes envoyées aux périphériques, en lecture ou en écriture. Suivant l'adresse décodée dans une requête et le sens du transfert (lecture ou écriture),

le décodeur d'adresses 24 sélectionne un motif de filtrage dans la mémoire locale 25 (ladite mémoire locale étant initialisée par le système d'exploitation 12 à la mise en route du système informatique).

5 La donnée comprise dans la requête est envoyée au filtre programmable 26 relié à la mémoire locale 25 et dans lequel le motif de filtrage est appliqué à cette donnée. Ce motif de filtrage indique au filtre 26 la fonction à appliquer individuellement sur chacun des bits de la donnée. Ainsi, le filtre 26 peut soit laisser inchangé le bit, soit le forcer à zéro, soit le forcer à un.

10 Chacun de ces motifs de filtrage constitue un critère de vérification d'intégrité du système.

La donnée modifiée issue du filtre programmable 26 est ensuite envoyée à l'interface de bus entrée-sortie 23 reliée au bus 2 des périphériques et indirectement aux périphériques.

15 Le dispositif selon l'invention comporte également une partie logicielle, qui vient compléter la partie matérielle formée par le module 16.

Cette partie logicielle est nécessaire pour exploiter correctement les fonctionnalités du module 16 par les applications 13, 14 et le système d'exploitation 12.

20 A cet effet, lors de l'initialisation du système exécuté sur le processeur central 3, le système d'exploitation 12 installe dans l'espace de mémoire virtuelle 130,140 (figure 5) de l'application 13, 14 un accès 133, 143 à l'espace d'adresses physiques 170 dans une zone particulière de décodage 131, 141 du module 16, dite zone d'io-pages virtuelles. Ceci est effectué pour chaque application 13, 14 susceptible de demander l'accès en lecture ou en
25 écriture à un périphérique particulier (par exemple unité de disque 9A). On note à titre de clarification que les adresses mémoires 133 et 143 correspondent aux adresses mémoires virtuelles propres à chaque application, alors que les zones 131 et 141 sont visibles dans l'espace d'adressage physique pour le processeur 3. Ainsi, les io-pages virtuelles de l'interface de périphérique 9A se retrouvent
30 dans l'espace mémoire physique 131,141 autant que dans l'espace de mémoire virtuelle 130 de l'application 1 dans la zone 133, et dans l'espace de mémoire virtuelle 140 de l'application 0 dans la zone 143.

La taille des zones d'io-pages virtuelles 131, 141 est équivalente à la taille de l'espace mémoire occupé normalement par l'interface physique 9A du périphérique 9 en question et contenant les io-pages physiques.

5 L'application 13, 14 vient lire et écrire les données échangées avec l'interface physique dans les io-pages 131, 141.

Le système d'exploitation 12 initialise ensuite, pour chaque application 13, 14, deux champs de vecteurs 160, 161 du module 16, spécifiant la corrélation entre les adresses des io-pages virtuelles 131, 141 et les adresses des io-pages physiques du périphérique 9A.

10 Enfin, le système d'exploitation 12 initialise pour chaque application une zone 133, 143 de la mémoire locale 25 du module 16, correspondant respectivement aux io-pages virtuelles 131, 141, avec les motifs de filtrage (le nombre de motifs de filtrage variant selon les périphériques) à appliquer à chaque accès de l'application 13, 14.

15 On note que les motifs sont partiellement identiques pour un même périphérique. Lors du filtrage, les adresses DMA (paramètres des requêtes) sont par exemple filtrées différemment pour tenir compte du schéma de translation des adresses virtuelles / adresses physiques de chaque application pour un même périphérique. Par contre, les motifs des registres de
20 contrôle seront potentiellement identiques.

A titre de clarification, la figure 5 représente alors un exemple d'espace d'adressage dans un système informatique où résident deux applications 13, 14 partageant un même périphérique 9.

Le système d'exploitation 12 peut accéder, par son espace virtuel
25 120,

- d'une part, aux champs de vecteurs 160, 161 (du module 16), et aux deux champs de bits 132, 142 comportant les motifs de filtrage des io-pages virtuelles 131, 141 pour chacune des deux applications 13, 14 (partageant le même périphérique 9) respectivement

30 - et, d'autre part, aux registres contenus dans l'interface physique 9A du périphérique 9.

Chaque application 13, 14 n'a accès, par son espace de mémoire

virtuelle 130, 140, en plus de la mémoire centrale habituelle pour son exécution (non représentée), qu'à la zone spécifique décodée 131, 141 par le module 16.

Sur la figure 5, la valeur N correspond à la taille de l'interface physique 9A accessible par le processeur central 3.

5 La valeur M correspond à la taille du champ de bits nécessaire pour coder la globalité du filtre correspondant à l'interface physique. Par exemple, si la taille de l'interface physique est de 64 octets, soit $64 \times 8 = 512$ bits, alors ce sont chacun de ces 512 bits d'io-pages physiques que le module 16 doit filtrer. En s'appuyant sur un exemple illustré par la figure 7, chaque bit peut
10 être soit laissé inchangé, soit forcé à 1, soit forcé à 0. Un filtre élémentaire de bit doit donc avoir au moins un de ces trois comportements. En codant le comportement lui même sous forme binaire, il occupe 2 bits (4 valeurs). Finalement, la taille N nécessaire pour coder l'ensemble des 512 filtres de bits est de $512 \times 2 = 1024$ bits, soit $1024 / 8 = 128$ octets.

15 La valeur K correspond à la taille du champ de vecteurs nécessaire pour décrire la translation entre les adresses des io-pages virtuelles 131,141 et les adresses des io-pages physiques 9A. Etant donné le nombre élevé de possibilités de codage du moyen de translation, nous nous limiterons à n'évoquer que les deux extrêmes :

- 20 - soit les translations sont fixes et non modifiables et, dans ce cas, la taille du champ de vecteurs est nulle ;
- soit la translation de chaque adresse d'io-page virtuelle est décrite individuellement. En appelant q la taille du bus d'adresses physiques (3,4,5 octets ou plus), chaque champ de vecteurs occupe
25 alors un espace de $K = q \times M$ octets. En reprenant l'exemple de l'interface contenant 64 octets d'io-pages, la taille du champ de vecteurs occupe donc sur un bus 32 bits (4 octets) $4 \times 64 = 256$ octets.

En ce qui concerne le procédé de contrôle d'accès aux périphériques, il comprend alors les étapes suivantes, illustrées par la figure 6.

30 Dans une première étape E1, comme on vient de le voir, lors de la mise en route du système informatique, le système d'exploitation 12 initialise la mémoire locale 25 du module 16 en lui envoyant les motifs de filtrage, à

appliquer aux adresses des io-pages concernées du périphérique partagé (ces motifs de filtrage étant préalablement récupérés dans une mémoire du système sous forme de champ de bits par exemple).

5 Le module 16 attend ensuite dans une étape E2 de recevoir une requête d'une application 13, 14 en lecture ou en écriture aux adresses des io-pages virtuelles 131,141. Cette requête est destinée au périphérique partagé 9.

10 Dans le cas d'une commande d'écriture provenant de l'unité centrale de traitement 3 (commande envoyée par une application 13, 14 exécutée par l'unité centrale de traitement 3), la donnée est modifiée dans une étape E3 conformément à ce qui a été exposé. La donnée est ensuite appliquée sur le bus d'adresses et de données 2 du côté des périphériques dans une étape E4, via l'interface 23, afin de transmettre la donnée vers l'io-page physique correspondante.

15 Dans le cas d'une commande en lecture sur une io-page virtuelle, la requête de lecture est transmise au périphérique dans une étape E5 afin de lire l'io-page physique correspondante. Puis le dispositif 16 attend une réponse dudit périphérique dans une étape E6. La donnée à modifier est alors celle provenant du bus 2 du côté des périphériques. Cette donnée est alors modifiée dans une étape E7, puis la donnée une fois modifiée est appliquée sur le bus
20 du processeur 2' au niveau de l'unité centrale de traitement 3, dans une étape E8.

Dans l'exemple déjà cité de la figure 7, le module est mis en œuvre, notamment, pour garantir cette intégrité lors de transfert de blocs par le mécanisme de DMA 102 proposé par le périphérique 9. Lors de l'initialisation
25 de l'application 13, celle-ci demande au système l'accès virtuel au périphérique 9, en demandant une zone d'adresses virtuelles 103 qu'elle va utiliser pour transférer des données, 64 kilo octets dans l'exemple. Connaissant les adresses physiques 105 de la zone de transfert 103 de l'application 13, le système d'exploitation exécuté sur le processeur 3 initialise le motif de filtrage
30 du registre d'adresse de DMA dans le champ de bits 132 en conséquence : sachant que la zone tampon 105 de l'application 13 s'étend de 0x0010.0000 (en base hexadécimale) à 0x0010.FFFF, le motif de filtrage doit laisser

inchangés les bits 0 à 15, forcer les bits 16 à 19 et 21 à 31 à zéro et le bit 20 à un. Ainsi, quelle que soit la valeur écrite par l'application 13 dans le registre DMA de l'interface virtuelle 133, la valeur envoyée à l'interface physique sera toujours dans la zone 105 appartenant à l'application 13.

5 On comprend qu'on a bien mis en place un dispositif de partage de périphériques, garantissant l'intégrité du système lors des accès en lecture ou en écriture à partir de et vers les périphériques dans tous les types de registres de ces périphériques. Il est clair que les applications doivent être modifiées pour tirer parti de l'accès direct aux périphériques qui leur est alors
10 offert, et pour pouvoir envoyer des commandes sans aucune abstraction, au plus près des commandes exécutables par chaque périphérique.

On a bien un dispositif principalement matériel, dans la mesure où, en dehors de l'initialisation de la mémoire locale 25 du module 16 par le système d'exploitation 12, les autres tâches de traitement des accès vers les
15 périphériques sont totalement prises en charge par le module 16 suivant une logique câblée par exemple.

Ce procédé permet d'obtenir une protection des accès jusqu'à la granularité du bit, puisqu'il est capable de surveiller chaque accès élémentaire. De plus, il permet de décharger les cycles CPU (de l'unité centrale de
20 traitement 3) du système d'exploitation 12, en déléguant la vérification des données transmises par le module 16.

En variante, l'unité de logique programmable 22 est insérée dans chacun des périphériques partagés, au lieu d'être insérée dans l'interface PCI
primaire.

25 Dans une variante de réalisation de type hybride comportant à la fois un dispositif matériel et logiciel et illustrée par la figure 8, le dispositif se présente sous la forme d'une logique plus souple, en terme de programmabilité.

Le module 16 comporte alors un processeur local 28 et une mémoire locale 25, connectés sur le bus processeur 2'.

30 De façon résumée, le processeur local 28 scrute tous les accès des applications 13, 14 en lecture comme en écriture sur les io-pages virtuelles 131, 141 pour leur faire subir un traitement de filtrage éventuel avant de

propager l'accès directement sur l'interface physique 9A dans les zones décodées.

Dans cette variante, la partie logicielle exécutée par le système d'exploitation 12 reste identique à celle de la réalisation matérielle exposée plus haut (figure 4).

De même, le principe de fonctionnement reste identique à celui exposé à la figure 6 et lors de la description faite en référence à cette figure.

On comprend que par ailleurs, en plus d'assurer la fonction principale de filtrage, la présence d'un processeur local 28 est une opportunité pour décharger l'unité centrale de traitement 3 de certaines tâches simples, telles que l'acquittement de requêtes par exemple.

Selon encore une autre variante de réalisation, purement logicielle cette fois, le dispositif utilise l'unité de gestion de mémoire (MMU) de l'unité centrale de traitement 3, connue de l'homme du métier. On rappelle que l'unité de gestion de mémoire MMU permet d'associer des pages, donc des adresses virtuelles, vues par les applications 13, 14, à des pages physiques, réellement présentes dans la mémoire vive 4.

Le dispositif repose uniquement sur l'utilisation du système de pagination mémoire présent dans la plupart des processeurs (unité centrale de traitement 3) et des erreurs associées. Il ne requiert aucun dispositif externe à ajouter au système informatique.

Détournant l'utilisation du mécanisme de translation de pages virtuelles en pages physiques, il est alors possible d'émuler le mécanisme d'io-pages virtuelles. En effet, en traitant de manière particulière les exceptions générées par l'unité de gestion de mémoire MMU lors des accès par les applications 13, 14 aux adresses d'io-pages virtuelles 133, 143, il est possible de contrôler les données lues et écrites par ces applications à partir de et vers les périphériques.

Bien entendu, la présente invention ne se limite pas aux détails des formes de réalisation décrits ici à titre d'exemple, mais s'étend au contraire aux modifications à la portée de l'homme de l'art.

REVENDEICATIONS

1. Dispositif de partage et de contrôle d'accès de périphérique pour un système informatique comprenant un processeur central (CPU) et au moins un périphérique d'entrée-sortie comportant une interface physique de contrôle accessible au processeur central, caractérisé en ce que ledit dispositif comporte :
- des moyens de reproduction fidèle sous forme d'interface virtuelle de l'interface physique d'au moins un périphérique,
 - des moyens d'interception par ladite interface virtuelle de toutes les requêtes et données échangées entre le processeur central et le périphérique, commandées par une application prédéterminée exécutée dans le système,
 - des moyens de modification éventuelle desdites requêtes et données interceptées selon au moins un critère prédéterminé.
2. Dispositif selon la revendication 1, caractérisé en ce que les moyens de reproduction sous forme virtuelle de cette interface physique (9A) comprennent :
- un espace de mémoire (131,141) réservé à l'image de l'interface physique, propre à chaque application exécutée par le système informatique,
 - un moyen permettant de lier les adresses de ces espaces de mémoire (131,141) à l'adresse de l'interface physique 9A.
3. Dispositif selon l'une quelconque des revendications 1 à 2, caractérisé en ce que les moyens d'interception comprennent :
- d'une part, une interface (21) avec le bus (2') relié à l'unité centrale de traitement (3), et une interface (23) avec le bus (2) relié aux périphériques (6, 9, 10),
 - et, d'autre part, un moyen de décodage d'adresses (24).
4. Dispositif selon l'une quelconque des revendications 1 à 3, caractérisé en ce que les moyens de modification comprennent un moyen de filtrage des requêtes interceptées par les moyens d'interception, selon au moins

un critère mémorisé dans un moyen de mémoire (25).

5. Dispositif selon l'une quelconque des revendications 1 à 4, caractérisé en ce qu'il se compose :

- d'un module (16) inséré entre l'unité centrale de traitement (3) et le bus des périphériques (2),
- et d'un élément logiciel stocké préalablement dans un moyen de mémoire de l'unité centrale de traitement (3).

6. Dispositif selon la revendication 5, caractérisé en ce que le module (16) comporte :

- une interface de bus entrée-sortie (21) reliée par le bus processeur (2') au couple formé de l'unité centrale de traitement (3) et de la mémoire (4) par l'intermédiaire du pont (19),
- une unité de logique programmable (22),
- une interface de bus entrée-sortie (23) reliée au bus d'adresses et de données (2).

7. Dispositif selon l'une quelconque des revendications 5 à 6, caractérisé en ce que l'unité de logique programmable (22) comporte un décodeur d'adresses (24), une mémoire locale (25) et un filtre programmable (26).

8. Dispositif selon la revendication 7, caractérisé en ce qu'il comporte des moyens d'insertion dans l'interface du bus de communication primaire (2') connecté à la mémoire vive (4).

9. Dispositif selon l'une quelconque des revendications 7 à 8, caractérisé en ce que le décodeur d'adresses (24) comporte des moyens de sélectionner au moins un motif de filtrage des données comprises dans une requête, selon l'adresse décodée dans la requête.

10. Dispositif selon l'une quelconque des revendications 7 à 9, caractérisé en ce que le filtre programmable (26) comporte des moyens adaptés à appliquer aux données comprises dans les requêtes des motifs de filtrage prédéterminés constituant des critères de vérification d'intégrité du système.

11. Dispositif selon l'une quelconque des revendications 5 à 10,

caractérisé en ce qu'il comporte des moyens adaptés à ce que, lors de l'initialisation du système, pour chaque application (13, 14) susceptible de demander l'accès en lecture ou en écriture à un périphérique particulier (9A), présent dans le système d'exploitation (12) en aval du module (16), le système d'exploitation (12) installe dans l'espace de mémoire virtuelle (130) de l'application (13, 14) un accès (133, 143) à la mémoire physique (4) dans une zone particulière (131, 141) appelée zone d'io-pages virtuelles du module (16).

12. Dispositif selon la revendication 11, caractérisé en ce que la taille de la zone d'io-pages virtuelles (131, 141) est équivalente à l'espace mémoire occupé par l'interface physique (9A) du périphérique (9) en question.

13. Dispositif selon l'une quelconque des revendications 11 à 12, caractérisé en ce qu'il comporte des moyens adaptés à ce que le système d'exploitation (12) initialise, pour chaque application (13, 14), un champ de vecteurs (160,161) spécifique à chaque application dans la mémoire locale (25) du module (16), spécifiant les adresses de translation des io-pages virtuelles (131,141) en io-pages physiques qui sont intégrées à l'interface physique du périphérique (9A).

14. Dispositif selon l'une quelconque des revendications 11 à 13, caractérisé en ce qu'il comporte des moyens adaptés à ce que le système d'exploitation (12) initialise, pour chaque application, une zone (132, 142) de la mémoire locale (25) du module (16), équivalente à la zone de décodage (131, 141), avec les motifs de filtrage à appliquer à chaque accès de l'application (13, 14).

15. Dispositif selon la revendication 14, caractérisé en ce qu'il comporte des moyens adaptés à ce que

- lors de la mise en route du système informatique, le système d'exploitation (12) initialise la mémoire locale (25) du module (16) en lui envoyant
 - o les motifs de filtrage, à appliquer aux différentes adresses d'io-pages virtuelles en lecture ou écriture pour les périphériques partagés,
 - o la translation entre les adresses des io-pages virtuelles

(141,131) et celles des io-pages physiques correspondantes dans l'interface physique (9A)

- le module (16) attende de recevoir une requête d'une application (13, 14) en lecture ou en écriture vers les périphériques partagés aux adresses d'io-pages virtuelles (131,141),
- dans le cas d'une commande d'écriture provenant de l'unité centrale de traitement (3), la donnée soit modifiée puis appliquée sur le bus d'adresses et de données (2) du côté des périphériques ,
- dans le cas d'une commande en lecture, la requête soit transmise au périphérique, puis le module (16) attende une réponse dudit périphérique, la donnée à modifier étant alors celle provenant du bus (2) du côté des périphériques, cette donnée soit alors modifiée, puis la donnée une fois modifiée soit appliquée sur le bus du processeur (2') au niveau de l'unité centrale de traitement (3).

16. Procédé de partage et de contrôle d'accès de périphérique pour un système informatique comprenant un processeur central (CPU) et au moins un périphérique d'entrée-sortie comportant une interface physique de contrôle accessible au processeur central, caractérisé en ce qu'il comporte :

- une étape de reproduction sous forme d'interface virtuelle de l'interface physique d'au moins un périphérique,
- une étape d'interception par ladite interface virtuelle de toutes les requêtes et données échangées entre le processeur central et le périphérique, commandées par une application prédéterminée exécutée dans le système,
- une étape de modification éventuelle desdites requêtes et données interceptées selon au moins un critère prédéterminé.

17. Procédé selon la revendication 16, caractérisé en ce que l'étape de reproduction sous forme virtuelle de cette interface physique (9A) comprend la création :

- d'un espace de mémoire (131,141) réservé à l'image de l'interface physique (9A), propre à chaque application exécutée par le système informatique,

- d'un mécanisme permettant de lier les adresses physiques (131,141) de ces espaces de mémoire à l'adresse de l'interface (9A)
- d'un champ (132) spécifiant les fonctions de filtrage à appliquer à la zone de mémoire (131).

5 18. Procédé selon l'une quelconque des revendications 16 à 17, caractérisé en ce qu'il comprend une étape de sélection d'au moins un motif de filtrage des données comprises dans une requête, selon l'adresse décodée dans la requête.

10 19. Procédé selon la revendication 18, caractérisé en ce qu'il comporte une étape d'application aux données comprises dans les requêtes des motifs de filtrage prédéterminés constituant des critères de vérification d'intégrité du système.

15 20. Procédé selon l'une quelconque des revendications 16 à 19, caractérisé en ce qu'il comporte une étape, lors de l'initialisation du système, pour chaque application (13, 14) susceptible de demander l'accès en lecture ou en écriture à un périphérique particulier (9A), présent dans le système d'exploitation (12) en aval du module (16), d'installation par le système d'exploitation (12) dans l'espace de mémoire virtuelle (130) de l'application (13, 14) d'un accès (133, 143) à la mémoire physique (4) dans une zone particulière
20 (131, 141) dite zone de décodage du module (16).

 21. Procédé selon la revendication 20, caractérisé en ce que la taille de la zone de décodage (131, 141) est équivalente à l'espace mémoire occupé par l'interface physique (9A) du périphérique (9) en question.

25 22. Procédé selon l'une quelconque des revendications 16 à 21, caractérisé en ce qu'il comporte une étape d'initialisation par le système d'exploitation (12), pour chaque application (13, 14), d'un champ de vecteurs (160,161) spécifique à chaque application dans la mémoire locale (25) du module (16), spécifiant les adresses de translation des io-pages virtuelles (131,141) en io-pages physiques qui sont intégrées à l'interface physique du
30 périphérique (9A).

 23. Procédé selon l'une quelconque des revendications 20 à 22, caractérisé en ce qu'il comporte une étape d'initialisation par le système

d'exploitation (12) pour chaque application d'une zone (132, 142) de la mémoire locale (25) du module (16), équivalente de la zone de décodage (131, 141), avec les motifs de filtrage à appliquer à chaque accès de l'application (13, 14).

24. Procédé selon la revendication 23, caractérisé en ce qu'il
5 comporte des étapes telles que :

- dans une première étape (E1), lors de la mise en route du système informatique, le système d'exploitation (12) initialise la mémoire locale (25) du module (16) en lui envoyant
 - 10 • les motifs de filtrage, à appliquer aux différentes adresses d'io-pages virtuelles en lecture ou écriture pour les périphériques partagés,
 - la translation entre les adresses des io-pages virtuelles (141,131) et celles des io-pages physiques correspondantes dans l'interface physique (9A),
- 15 - dans une étape (E2), le module (16) attende de recevoir une requête d'une application (13, 14) en lecture ou en écriture vers les périphériques partagés aux adresses d'io-pages virtuelles (131,141),
 - dans le cas d'une commande d'écriture provenant de l'unité centrale de traitement (3), la donnée soit modifiée dans une étape (E3) puis appliquée sur le bus d'adresses et de données (2) du côté des
 20 périphériques dans une étape (E4),
 - dans le cas d'une commande en lecture, la requête soit transmise au périphérique dans une étape (E5), puis le module (16) attende une réponse dudit périphérique dans une étape (E6), la donnée à modifier étant alors celle provenant du bus (2) du côté des
 25 périphériques, cette donnée soit alors modifiée dans une étape (E7), puis la donnée une fois modifiée soit appliquée sur le bus du processeur (2') au niveau de l'unité centrale de traitement (3), dans une étape (E8).

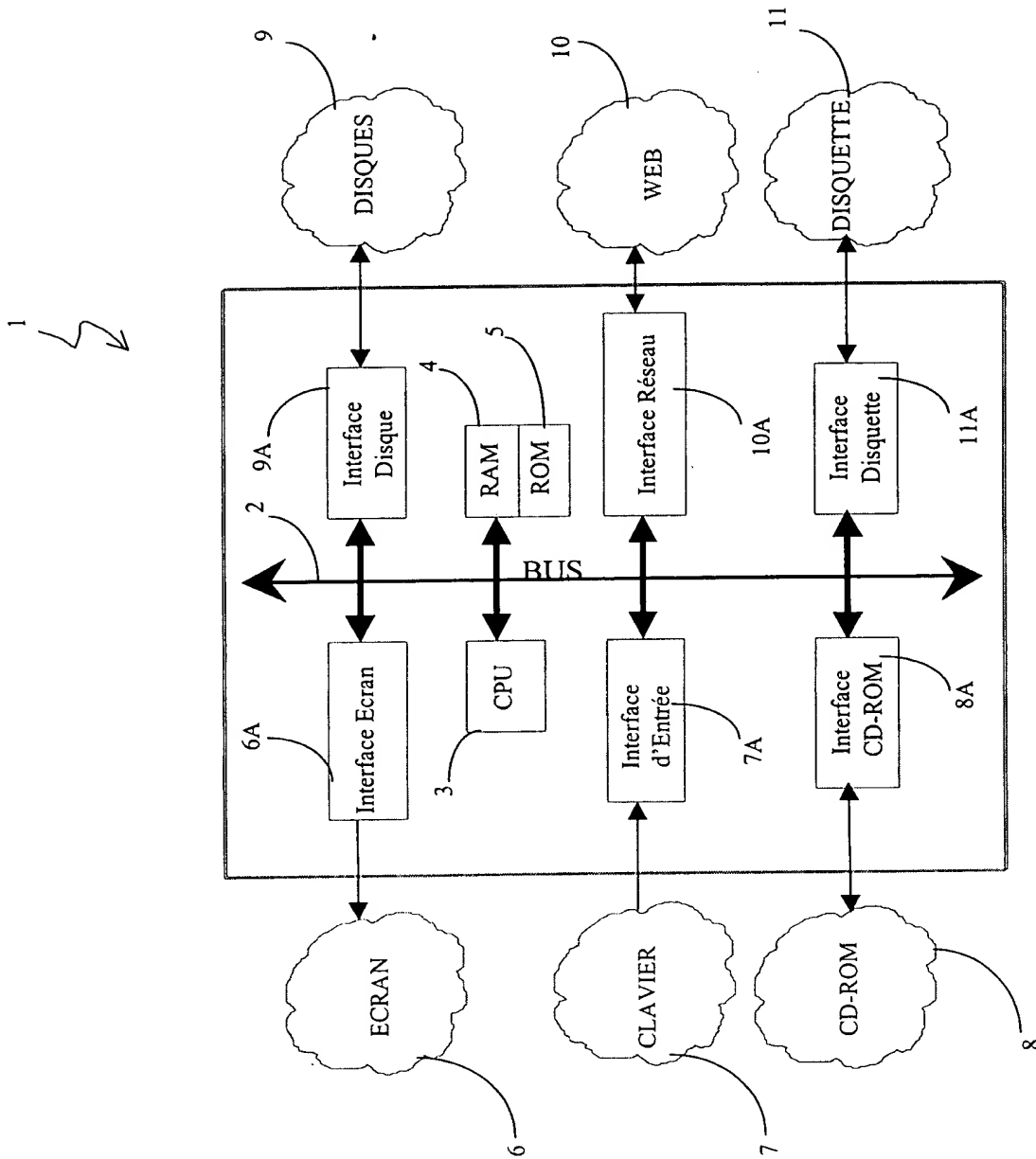


Fig.1

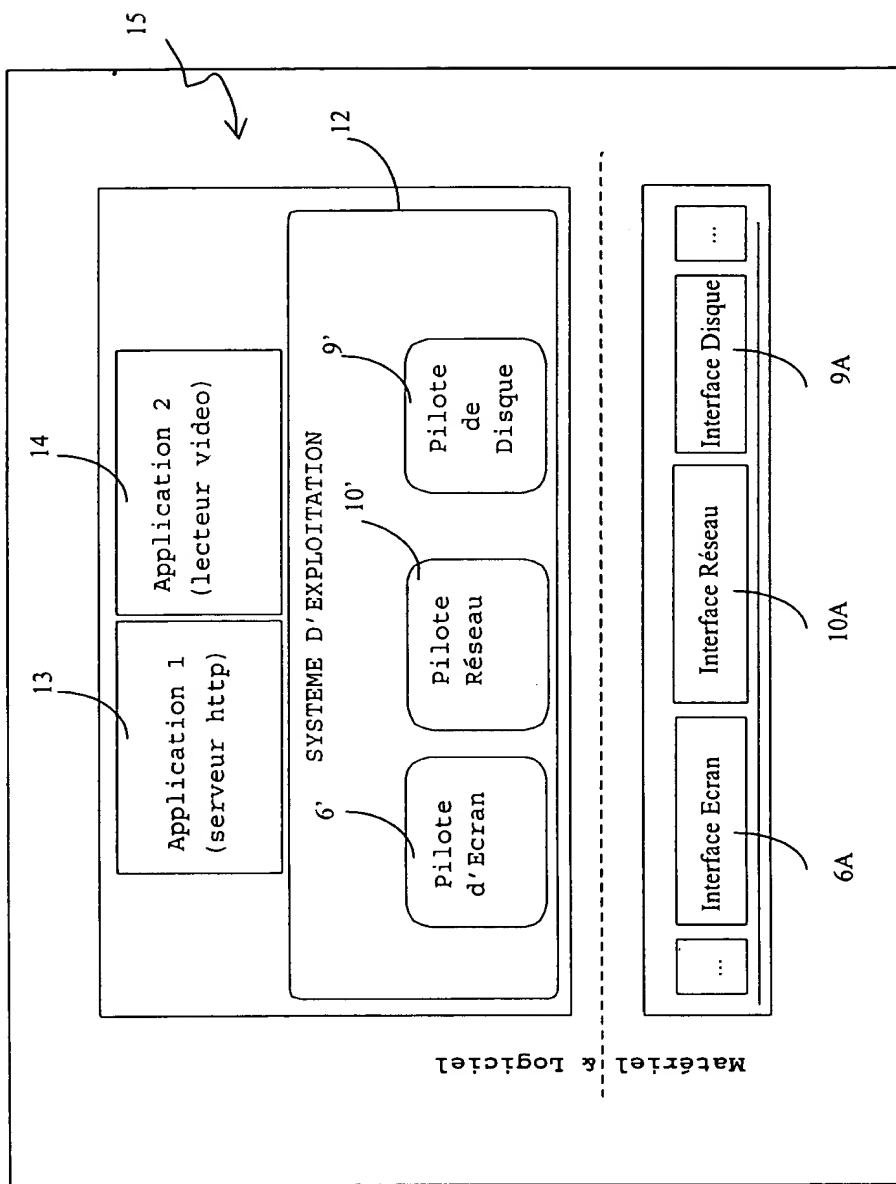


Fig.2

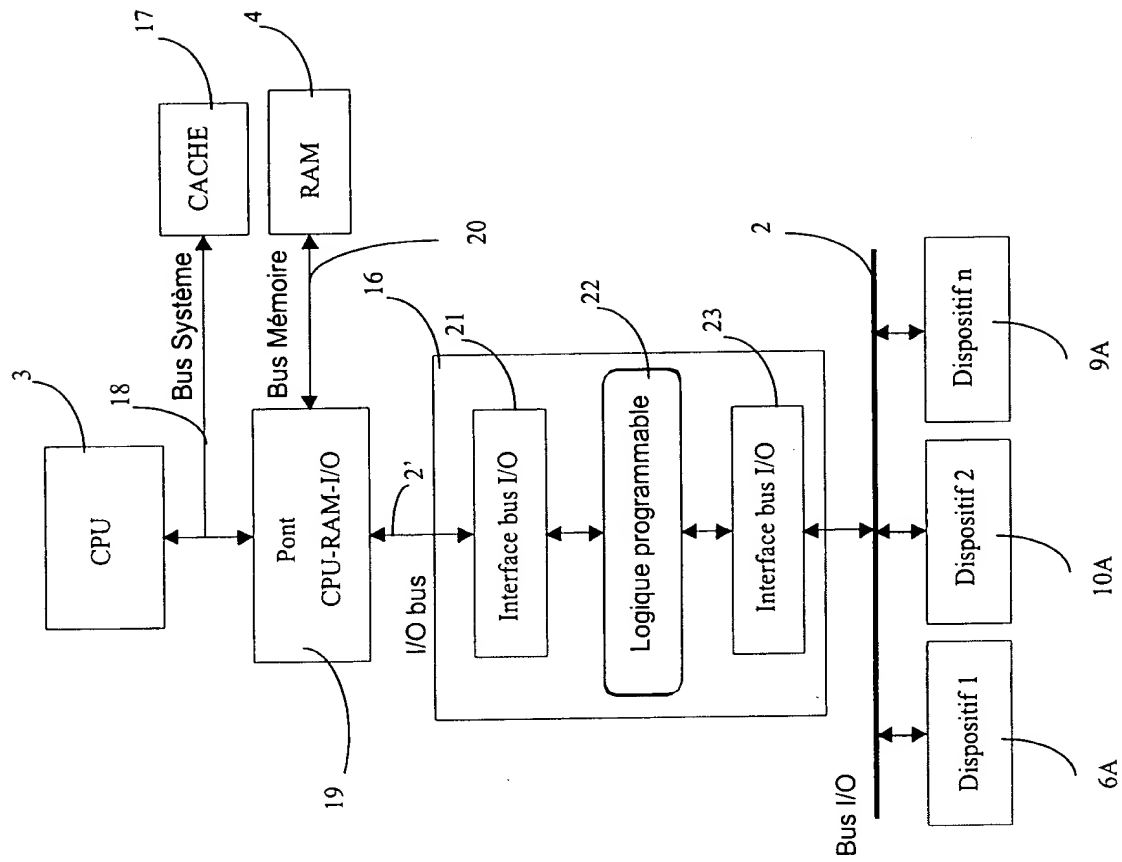


Fig.3

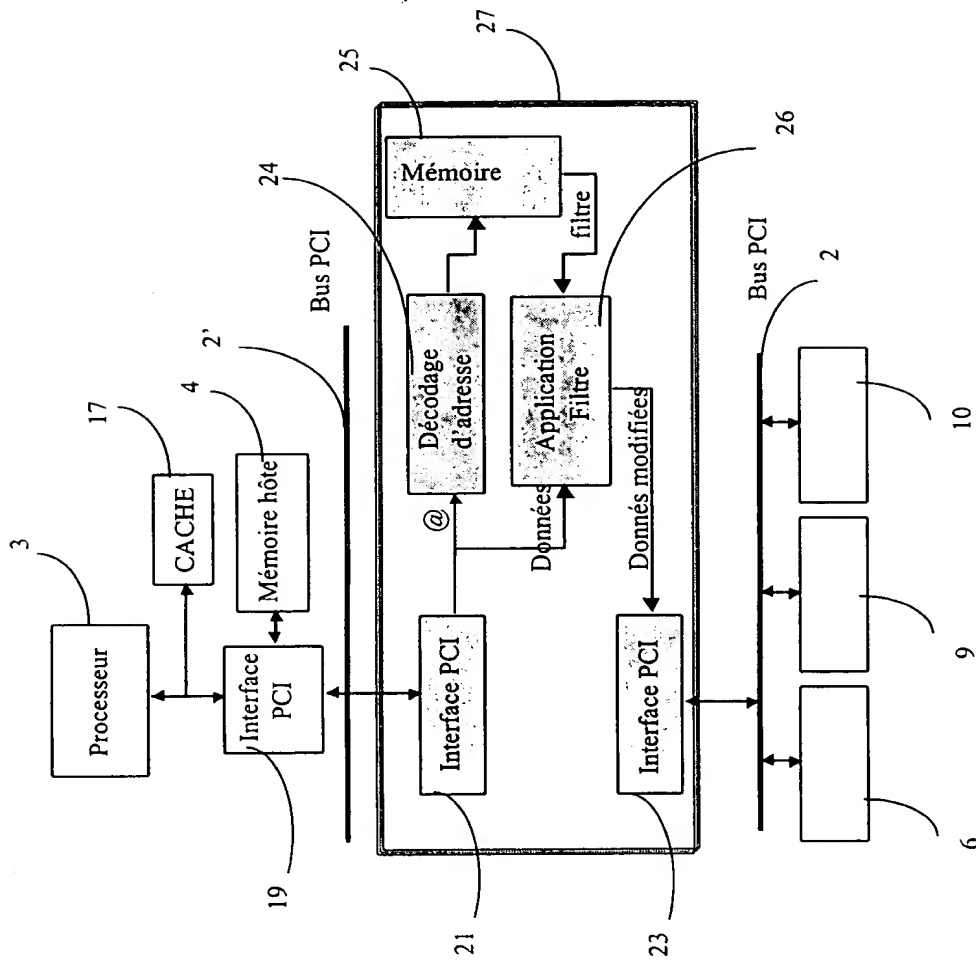


Fig.4

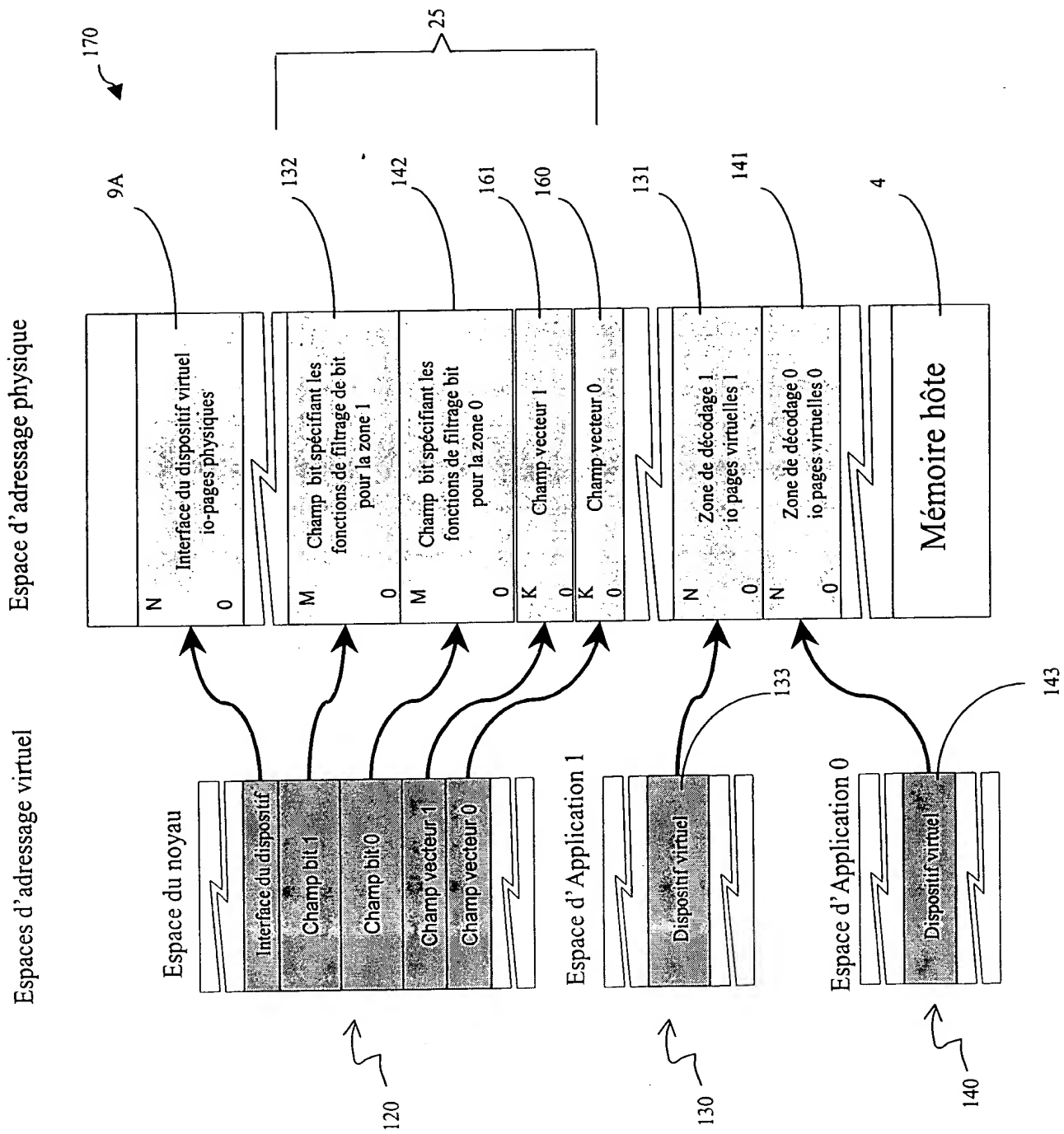


Fig.5

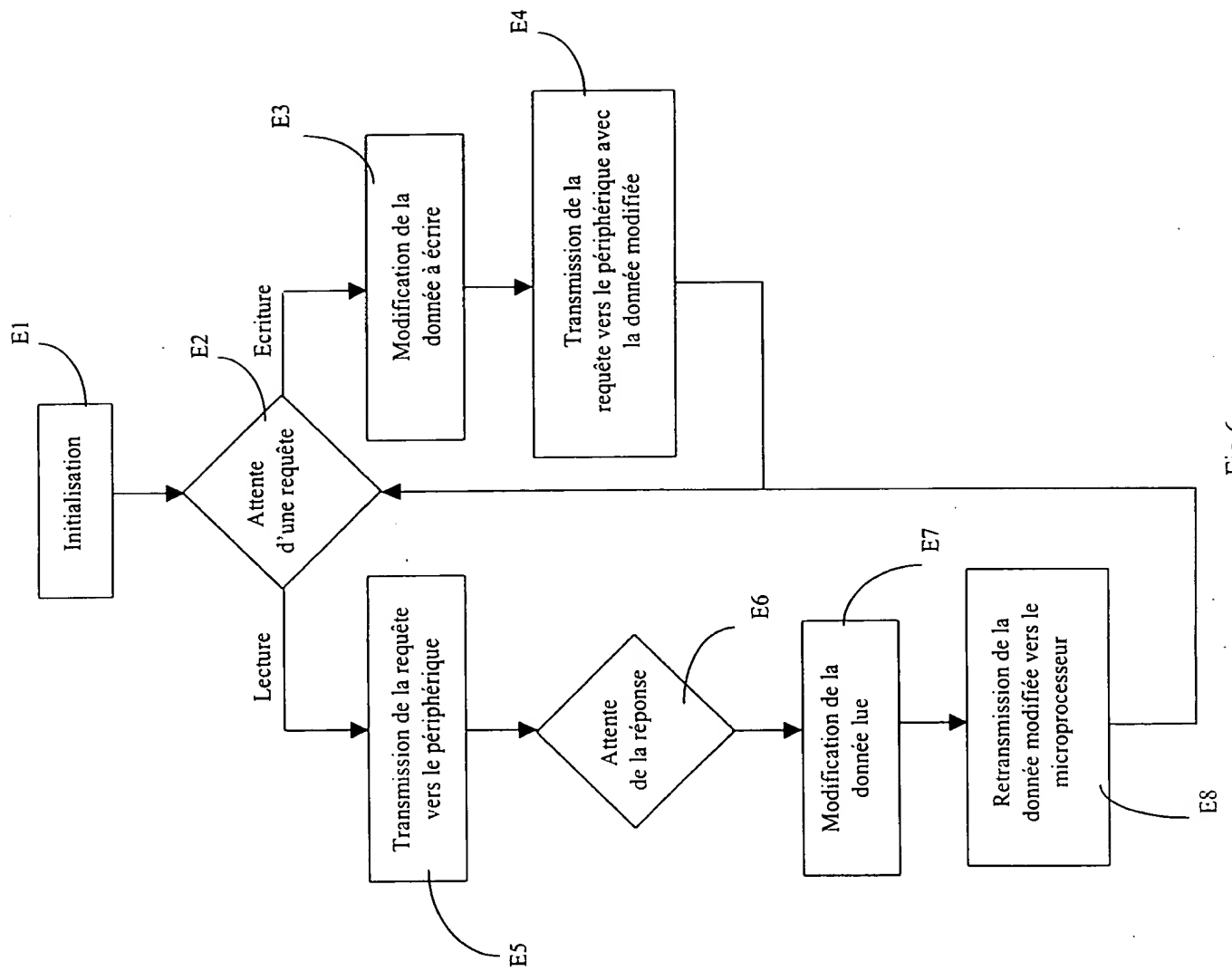


Fig. 6

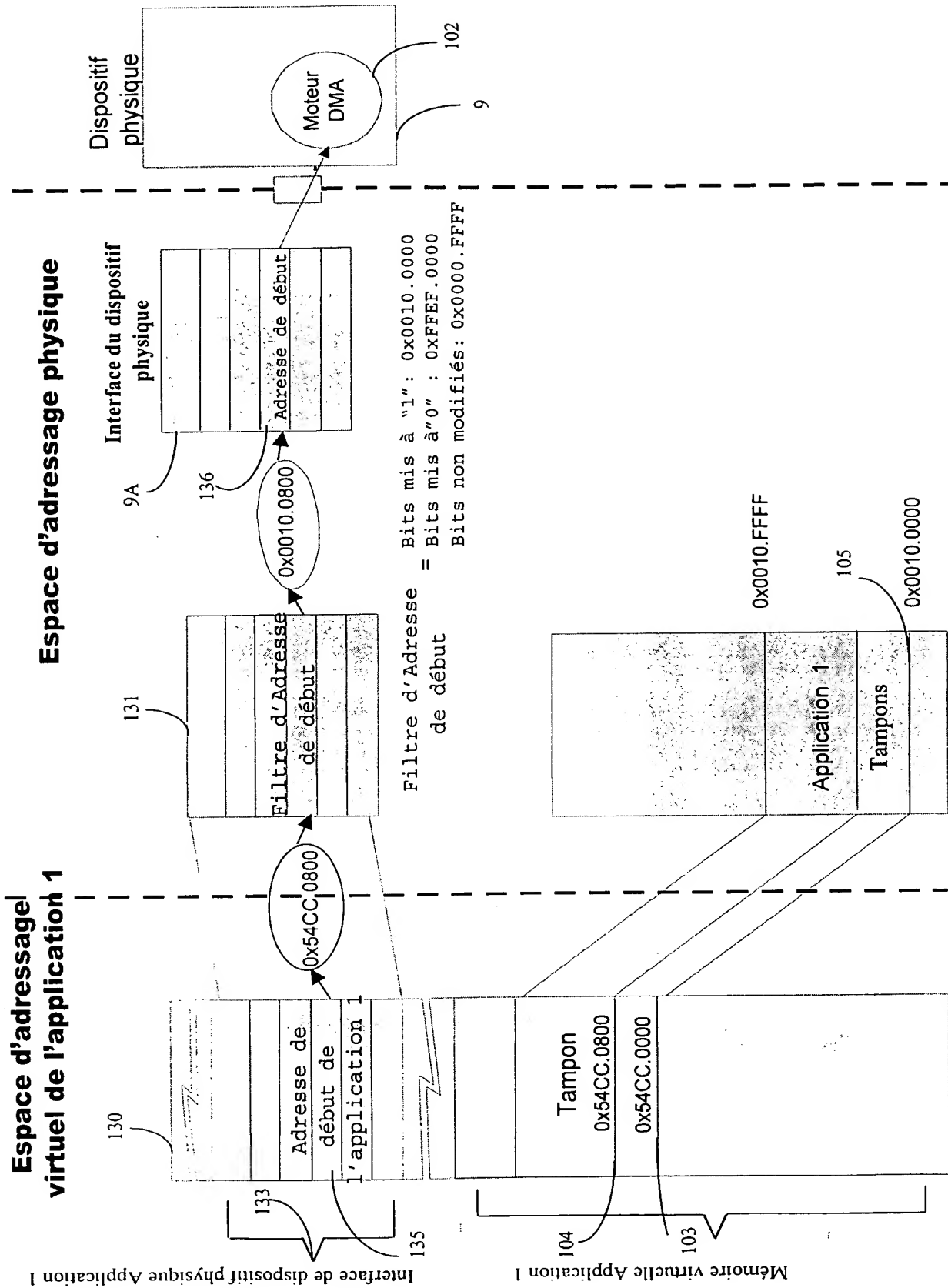


Fig.7

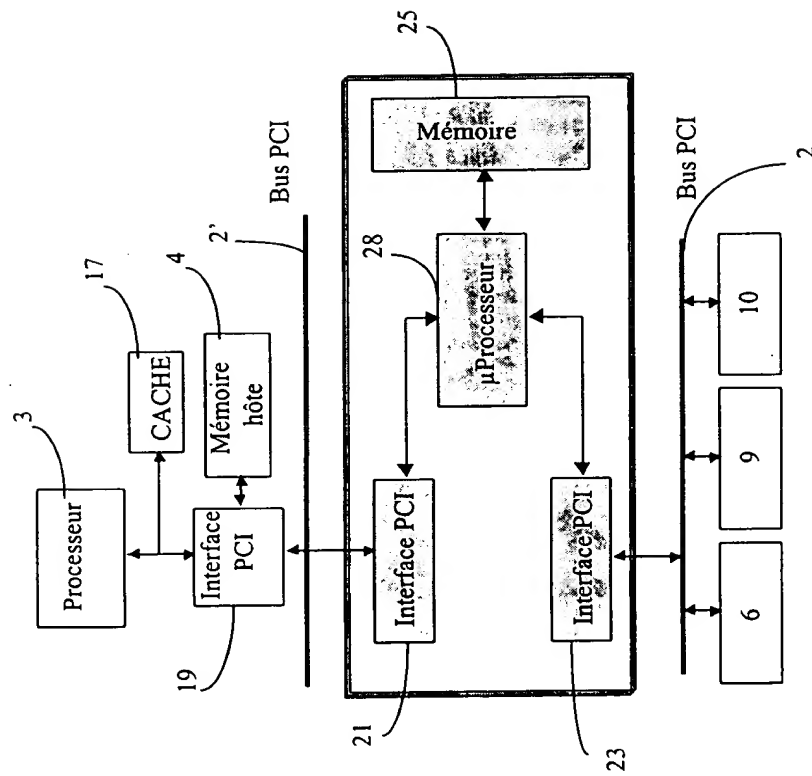


Fig.8

THIS PAGE BLANK (USPTO)